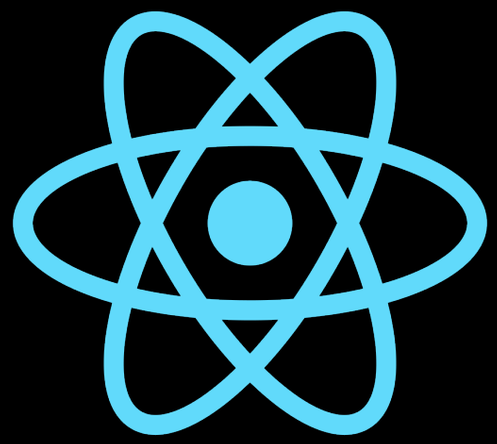


React

HSR

Sommer 2020

Alex Kühne

BA Web Development & Design
Eidg. FA Informatiker / Applikationsentwickler

swisscom

SIX

Swiss Re

Tages-Anzeiger

MIGROS

React

- wer hat davon gehört?
- wer hat es benutzt?
- wen interessiert’s? ;-)

Agenda

- Einführung in React
- Entwickeln mit React
- Test Driven Development
- Hooks & Tipps & Tricks
- *Zwischendurch: Praxis*

Was ist React?

„[one of] The three most popular frontend frameworks“

State of JavaScript 2019

- React
- Vue
- Angular

Was ist React?

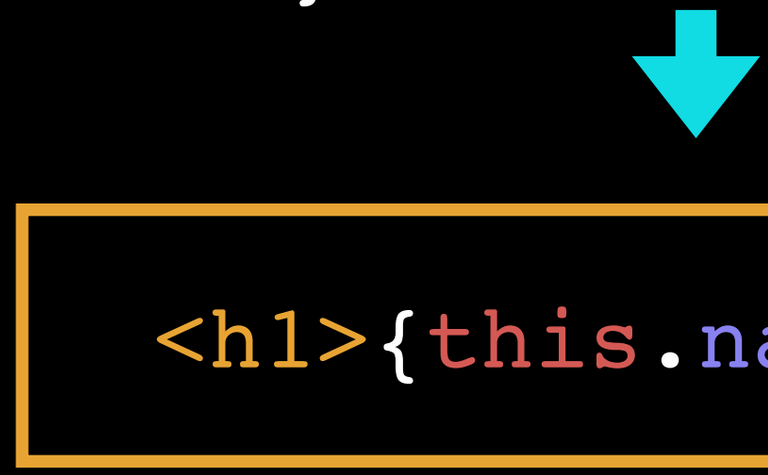
„A declarative, component-based JavaScript library for building user interfaces“

facebook.com

Was ist React?

Kurz gesagt: ein DOM-Renderer

```
{
  name: "Steve"
}
```




```
<h1>Steve</h1>
```

React Component

```
<h1>{this.name}</h1>
```

Name.js

```
<div>
  <Name />
</div>
```

```
<div>
  <h1>Steve</h1>
</div>
```

Vorteile von Komponenten?

Components…

…sind wiederverwendbar:

```
<div>
  <Name />
  <Name />
  <Name />
</div>
```

```
<div>
  <h1>Steve</h1>
  <h1>Steve</h1>
  <h1>Steve</h1>
</div>
```

Components…

…sind verschachtelbar:

```
<div>
  <Familie></Familie>
</div>
```

Components…

…sind verschachtelbar:

```
<div>
  <Familie>
    <Mutter />
    <Vater />
    <Kind />
  </Familie>
</div>
```

Components…

…sind verschachtelbar:

```
<div>
  <Familie>
    <Mutter />
    <Vater />
    <Kind>
      <Name />
    </Kind>
  </Familie>
</div>
```

Components…

…sind verschachtelbar:

```
<div>
  <Familie>
    <Mutter />
    <Vater />
    <Kind>
      <Name />
    </Kind>

    <Nachbar></Nachbar>
  </Familie>
</div>
```

Components…

…sind verschachtelbar (auch rekursiv):

```
<div>
  <Familie>
    <Mutter />
    <Vater />

    <Kind>
      <Name />
    </Kind>

    <Nachbar>
      <Familie />
    </Nachbar>
  </Familie>
</div>
```

HOC
(Higher Order Components)

Components…

…sind parametrisierbar:

Props

```
<div>
  <Name name="Steve" />
  <Name name="Bill" />
  <Name name="Elon" />
</div>
```

```
<div>
  <h1>Steve</h1>
  <h1>Bill</h1>
  <h1>Elon</h1>
</div>
```

Components im Detail

```
<Name name="Steve" />
```

```
class Name extends React.Component {
}
```

```
class Name extends React.Component {
  render() {
  }
}
```

```
class Name extends React.Component {
  render() {
    return <h1>{this.props.name}</h1>;
  }
}
```

```
class Name extends React.Component {
  render() {
    return <h1>{this.props.name}</h1>;
  }
}
```

HTML in JavaScript?!

```jsx
return <h1 id="title">Steve</h1>;
```

JSX Precompiler

```jsx
return React.createElement(
  "h1",
  { id: "title" },
  "Steve"
);
```

HTML in JS

Vorteile?

- alles in einem File (trotzdem auslagerbar)
- Typisierung auch in Templates:

```
<h1>{this.props.name}</h1>
// this.props does not exist
```

- Computed Styles:

```
<div style={{
  width: this.count * 10
}} />
```

Components im Detail

```
<Seconds />
```

```
class Seconds extends React.Component {
}
```

```
class Seconds extends React.Component {
  render() {
  }
}
```

```
class Seconds extends React.Component {
  render() {
    return <span>0</span>;
  }
}
```

```
class Seconds extends React.Component {
  render() {
    return <span>{second}</span>;
  }
}
```

Wo speichern?

Im State!

```
class Seconds extends React.Component {
  constructor() {
  }

  render() {
    return <span>{second}</span>;
  }
}
```

```
class Seconds extends React.Component {
  constructor() {
    this.state = {
      second: 0
    };
  }

  render() {
    return <span>{second}</span>;
  }
}
```

```
class Seconds extends React.Component {
  constructor() {
    this.state = {
      second: 0
    };
  }

  render() {
    return <span>{this.state.second}</span>;
  }
}
```

```
class Seconds extends React.Component {
  constructor() {
    this.state = {
      second: 0
    };

    setInterval(() => {
      // Erhöhe Sekunde
    }, 1000);
  }

  render() {
    return <span>{this.state.second}</span>;
  }
}
```

```
class Seconds extends React.Component {
  constructor() {
    this.state = {
      second: 0
    };

    setInterval(() => {
      // Erhöhe Sekunde
      this.setState({
      });
    }, 1000);
  }

  render() {
    return <span>{this.state.second}</span>;
  }
}
```

```
class Seconds extends React.Component {
  constructor() {
    this.state = {
      second: 0
    };

    setInterval(() => {
      // Erhöhe Sekunde
      this.setState({
        second: this.state.second + 1
      });
    }, 1000);
  }

  render() {
    return <span>{this.state.second}</span>;
  }
}
```

Summary

- Components
- Props
- State und `setState()`
- JSX

React im Detail

- 2013 entwickelt von und für facebook.com (besteht aus über 1’000 Komponenten)
- „erstes“ komponenten-basiertes Framework
- mit Abstand schnellstes Rendering ggü. damaliger Konkurrenz (AngularJS, Ember, …)

Doch warum ist React so schnell?

Virtual DOM Diff Algorithmus

Was ist nochmals DOM?

Virtual DOM Diff Algorithmus

VDOM

```
<span>0</span>
```

DOM

Virtual DOM Diff Algorithmus

VDOM

DOM

```
<span>0</span>
```

```
<span>0</span>
```

Virtual DOM Diff Algorithmus

VDOM

```
<span>1</span>
```

DOM

```
<span>0</span>
```

Virtual DOM Diff Algorithmus

VDOM

DOM

```
<span>1</span>
```

```
<span>1</span>
```

Virtual DOM Diff Algorithmus

VDOM

```
<span>2</span>
```

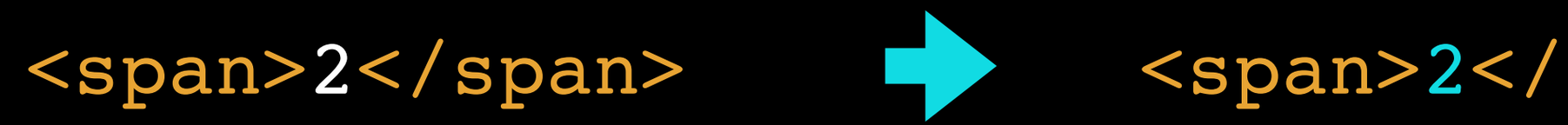
DOM

```
<span>1</span>
```

Virtual DOM Diff Algorithmus

VDOM

DOM



Auswirkungen

Vue und Ember 2+
Implementieren auch einen Virtual DOM

AngularJS -> Angular 2+
Komponentenbasiert,
Rendering-Boost durch unterschiedliche „Threads“ (Webworker) im Main-Thread

React 16 „Fiber“ (seit 2017)
Nach wie vor Virtual DOM, rendert aber jetzt mittels *requestAnimationFrame()* jetzt mit 60fps „native feeling“

Warum React?

…als Alternative zu z.B. Angular?

Facebook selbst benutzt React
Interesse, abwärtskompatibel zu bleiben

Source of innovation
Components, CLI, 60fps, Async, JSX, Hooks …

KISS
Straight-forward, einfache all-in-one Komponenten,
kein Ivy / JIT, etc., keine neue, proprietäre Syntax

Warum React?

…keine neue, proprietäre Syntax?

Templates ohne *ngIf und *ngFor?

…mit standard JavaScript Funktionen!

Erweitertes Templating

Loops (with `.map()`)

```
["Steve", "Bill", "Elon"].map(name =>
  "Mein Name ist " + name
);
```

```
[
  "Mein Name ist Steve",
  "Mein Name ist Bill",
  "Mein Name ist Elon"
]
```

Erweitertes Templating

Loops (with `.map()`)

```
["Steve", "Bill", "Elon"].map(name =>
  <Name name={name} />
);
```

```
[
  <Name name="Steve" />,
  <Name name="Bill" />,
  <Name name="Elon" />
]
```

Erweitertes Templating

if

```
isLoggedIn && <LogoutButton />
```

Erweitertes Templating

else

```
isLoggedIn || <LoginButton />
```

Erweitertes Templating

if / else

```
isLoggedIn
? <LogoutButton />
: <LoginButton />
```

Erweitertes Templating

IIFE

```
(() => {
    // Code hier

    return <MyComp />;
})()
```

Components im Detail

Lifecycle

```
class Name extends React.Component {
  constructor() {}

  render() {
    return <span>{this.props.name}</span>;
  }
}
```

```
class Name extends React.Component {
  constructor() {}

  render() {
    return <span>{this.props.name}</span>;
  }

  componentDidMount() / componentDidUpdate() {
    // Jetzt im DOM! (Benutze jQuery, etc…)
  }
}
```

```
class Name extends React.Component {
  constructor() {}

  render() {
    return <span>{this.props.name}</span>;
  }

  componentDidMount() / componentDidUpdate() {
    // Jetzt im DOM! Benutze jQuery, etc…
  }

  componentWillUnmount() {
    // Clean up, clearInterval, unbind events…
  }
}
```

```
class Name extends React.Component {
  constructor() {}

  shouldComponentUpdate() {
    return true; // or false
  }

  render() {
    return <span>{this.props.name}</span>;
  }

  componentDidMount() / componentDidUpdate() {
    // Jetzt im DOM! Benutze jQuery, etc…
  }

  componentWillUnmount() {
    // Clean up, clearInterval, unbind events…
  }
}
```

Components im Detail

Keyboard Input

```
class MyName extends React.Component {
  render() {
    return (
      <div>
        <input placeholder="My name" />
        <span>Hi Steve!</span>
      </div>
    );
  }
}
```

```
class MyName extends React.Component {
  constructor() {
    this.state = { name: "" };
  }

  render() {
    return (
      <div>
        <input placeholder="My name" />
        <span>Hi Steve!</span>
      </div>
    );
  }
}
```

```
class MyName extends React.Component {
  constructor() {
    this.state = { name: "" };
  }

  render() {
    return (
      <div>
        <input placeholder="My name" />
        <span>Hi {this.state.name}!</span>
      </div>
    );
  }
}
```

```
class MyName extends React.Component {
  constructor() {
    this.state = { name: "" };
  }

  render() {
    return (
      <div>
        <input placeholder="My name" onInput={e =>
          // e.target.value
        } />
        <span>Hi {this.state.name}!</span>
      </div>
    );
  }
}
```

```
class MyName extends React.Component {
  constructor() {
    this.state = { name: "" };
  }

  setName(name) { this.setState({ name }); }

  render() {
    return (
      <div>
        <input placeholder="My name" onInput={e =>
          // e.target.value
        } />
        <span>Hi {this.state.name}!</span>
      </div>
    );
  }
}
```

```
class MyName extends React.Component {
  constructor() {
    this.state = { name: "" };
  }

  setName(name) { this.setState({ name }); }

  render() {
    return (
      <div>
        <input placeholder="My name" onInput={e =>
          this.setName(e.target.value)
        } />
        <span>Hi {this.state.name}!</span>
      </div>
    );
  }
}
```

create-react-app

public/index.html

src/index.js

src/App.js

src/App.test.js

create-react-app

public/index.html

src/index.js

src/App.js

src/App.test.js

```
<html>
  <head>
    <title>React App</title>
  </head>
  <body>
    <div id="root"></div>

    <script src="bundle.js">
    </script>
  </body>
</html>
```

create-react-app

public/index.html

src/index.js

src/App.js

src/App.test.js

```
import React from "react";
import ReactDOM from "react-dom";
import App from "./App";

ReactDOM.render(
  <App />,
  document.getElementById("root")
);
```

create-react-app

public/index.html

src/index.js

src/App.js

src/App.test.js

```
import React from "react";
import Seconds from "./Seconds";

function App {
  return (
    //  Inhalt hier ersetzen durch
    //  Praxisübungen, z.B.:
    <div className="App">
      // <Seconds />
    </div>
  );
}
```

JSX Gotchas

- Keine self-closing Tags:

~~`<img>`~~

```
<img />
```

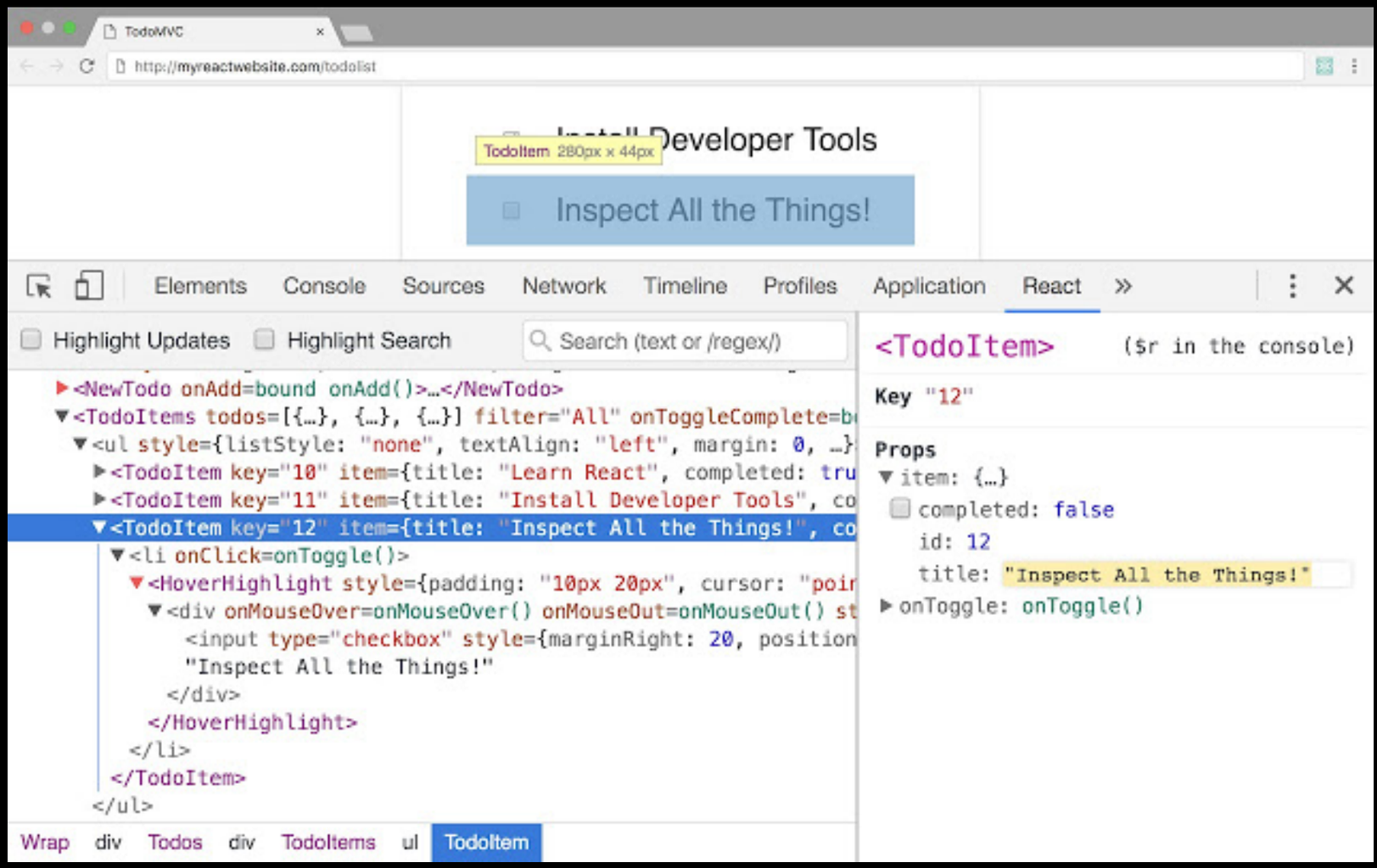
- `className` statt `class`:

~~`<div class="nice"></div>`~~

```
<div className="nice"></div>
```

React Developer Tools

Extension für Firefox und Chrome



Installation

React Developer Tools installieren

```
$ npx create-react-app hsr --typescript
```

`hsr/` in IDE öffnen

`src/App.js` „leeren“ (gesamten `<header />` löschen)

`src/Seconds.js` erstellen und einbinden (in `App.js`)
gilt für jede neue Komponente!

```
$ npm start / npm test
```

Praxisübung

Vorlage für jede neue Komponente

```
import React from "react";

class MyComponent extends React.Component {
  constructor() {
    super();
    // Code hier, z.B. this.state = {…};
  }

  render() { /* z.B. return <div></div>; */ }
}

export default MyComponent;
```

Praxisübung

`<Seconds />`

Praxisübung

`<MyName />`

Bonusaufgabe

Leeres `<span>Hi !</span>` nicht darstellen!

Components im Detail

Stateless Components

Stateless Components

Komponenten mit `function` anstatt `class`

```
function Name() {
  return <h1>Steve</h1>;
}
```

```
<Name />

// …oder mit ES6 (Arrow Function):
const Name = () => <h1>Steve</h1>;
```

Stateless Components

Komponenten mit `function` anstatt `class`

```
<Name name="Steve" /> ?
```

```
function Name(props) {
  return <h1>{props.name}</h1>;
}
```

Stateless Components

Wann und warum?

- Weniger Code als Klassen
- Komponenten, die „nur“ Elemente ausgeben
- Für Komponenten ohne „Innenleben“

Stateless Components

Real-world example

public/index.html

src/index.js

src/App.js

src/App.test.js

```
import React from "react";
import Seconds from "./Seconds";

function App() {
  return (
    <div className="App">
      <Seconds />
    </div>
  );
}
```

Components im Detail

Error Handling

```
<Name name="Steve" />
```

```
class Name extends React.Component {
  render() {
    return <span>{this.props.name}</span>;
  }
}
```

```
class Name extends React.Component {
  render() {
    console.log(x);

    return <span>{this.props.name}</span>;
  }
}
```

```
class Name extends React.Component {
  render() {
    console.log(x); // Error: x is not defined

    return <span>{this.props.name}</span>;
  }
}
```

```
<Name name="Steve" />
```

```
<ErrorHandler>
  <Name name="Steve" />
</ErrorHandler>
```

```
class ErrorHandler extends React.Component {
  render() {
    return this.props.children;
  }
}
```

```
class ErrorHandler extends React.Component {
  componentDidCatch(error) {
    // Ups… Fehlermeldung anzeigen!
  }

  render() {
    return this.props.children;
  }
}
```

Achtung: Catched nicht sich selbst!

Routing

```
import {
  BrowserRouter,
  Route,
  Link
} from "react-router-dom";
```

```
class App extends React.Component {
  render() {
    return (
      <h1>Mein Onlineshop</h1>
    );
  }
}
```

```
class App extends React.Component {
  render() {
    return (
      <BrowserRouter>
        <h1>Mein Onlineshop</h1>
      </BrowserRouter>
    );
  }
}
```

```
class App extends React.Component {
  render() {
    return (
      <BrowserRouter>
        <h1>Mein Onlineshop</h1>

        <Link to="/agb">AGB anzeigen</Link>
      </BrowserRouter>
    );
  }
}
```

```
class App extends React.Component {
  render() {
    return (
      <BrowserRouter>
        <h1>Mein Onlineshop</h1>

        <Link to="/agb">AGB anzeigen</Link>

        <Route path="/agb">
        </Route>
      </BrowserRouter>
    );
  }
}
```

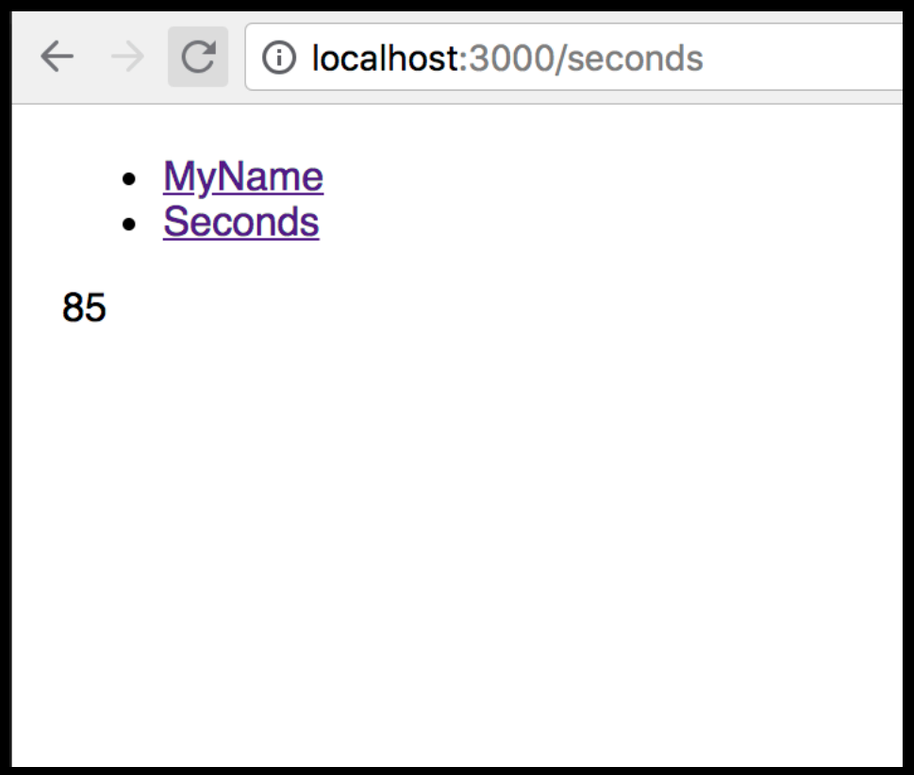
```
class App extends React.Component {
  render() {
    return (
      <BrowserRouter>
        <h1>Mein Onlineshop</h1>

        <Link to="/agb">AGB anzeigen</Link>

        <Route path="/agb">
          <ul>…</ul>
        </Route>
      </BrowserRouter>
    );
  }
}
```

Praxisübung

Routing mit `<BrowserRouter />`




```
$ npm install react-router-dom
```

Test Driven Development

Mit React

TDD Methode

Kurzrepetition

1. Test schreiben (vor der Implementation!)
2. Test ausführen → schlägt fehl
3. Implementation schreiben
4. Test ausführen → funktioniert!

TDD Vorteile

Kurzrepetition

- Man schreibt den Test garantiert
- Bisherige Funktionalität gewährleistet
- Test demonstriert API

create-react-app

public/index.html

src/index.js

src/App.js

src/App.test.js

```
import React from "react";
import { render }
  from "@testing-library/react";
import App from "./App";
```

```
test("renders…", () => {
  // …
  expect()….toBe()…
});
```

Test im Detail

```
<Seconds />
```

```
import React from "react";
import { render } from "@testing-library/react";
import Seconds from "./Seconds";
```

```
import React from "react";
import { render } from "@testing-library/react";
import Seconds from "./Seconds";

test("renders initial second", () => {
});
```

```
import React from "react";
import { render } from "@testing-library/react";
import Seconds from "./Seconds";

test("renders initial second", () => {
  const { getByText } = render(<Seconds />);
});
```

```
import React from "react";
import { render } from "@testing-library/react";
import Seconds from "./Seconds";

test("renders initial second", () => {
  const { getByText } = render(<Seconds />);
  expect(getByText("0")).toBeInTheDocument();
});
```

```
import React from "react";
import { render } from "@testing-library/react";
import Seconds from "./Seconds";

test("renders initial second", () => {
  const { getByText } = render(<Seconds />);
  expect(getByText("0")).toBeInTheDocument();
});
```

```
test("renders three seconds", () => {
});
```

```
import React from "react";
import { render } from "@testing-library/react";
import Seconds from "./Seconds";

test("renders initial second", () => {
  const { getByText } = render(<Seconds />);
  expect(getByText("0")).toBeInTheDocument();
});
```

```
test("renders three seconds", () => {
  jest.useFakeTimers();
});
```

```
import React from "react";
import { render } from "@testing-library/react";
import Seconds from "./Seconds";

test("renders initial second", () => {
  const { getByText } = render(<Seconds />);
  expect(getByText("0")).toBeInTheDocument();
});
```

```
test("renders three seconds", () => {
  jest.useFakeTimers();
  const { getByText } = render(<Seconds />);
});
```

```
import React from "react";
import { render } from "@testing-library/react";
import Seconds from "./Seconds";

test("renders initial second", () => {
  const { getByText } = render(<Seconds />);
  expect(getByText("0")).toBeInTheDocument();
});
```

```
test("renders three seconds", () => {
  jest.useFakeTimers();
  const { getByText } = render(<Seconds />);
  jest.runOnlyPendingTimers();
  jest.runOnlyPendingTimers();
  jest.runOnlyPendingTimers();
});
```

```
import React from "react";
import { render } from "@testing-library/react";
import Seconds from "./Seconds";

test("renders initial second", () => {
  const { getByText } = render(<Seconds />);
  expect(getByText("0")).toBeInTheDocument();
});
```

```
test("renders three seconds", () => {
  jest.useFakeTimers();
  const { getByText } = render(<Seconds />);
  jest.runOnlyPendingTimers();
  jest.runOnlyPendingTimers();
  jest.runOnlyPendingTimers();
  expect(getByText("3")).toBeInTheDocument();
});
```

Praxisübung

Test für `<Seconds />`

`src/App.test.js` löschen!

Praxisübung

Test für `<MyName />`

```
// Keyboard event simulieren

import { fireEvent, … } from "@testing-lib…";

const { getByPlaceholderText, … } = render(…);
const input = getByPlaceholderText("My name");

fireEvent.input(input, { target: { value:
"Steve" } });
```

Components im Detail

parent => child Kommunikation

```
class Parent extends React.Component {
  render() {
    return (
      <Child />
    );
  }
}
```

Components im Detail

parent => child Kommunikation

```
class Parent extends React.Component {
  render() {
    return (
      <Child
        myProp="myValue"
      />
    );
  }
}
```

Components im Detail

child => parent Kommunikation

```
class Parent extends React.Component {
  render() {
    return (
      <Child
        myProp="myValue"
        myCallback={(x) => …} // x von Kind
      />
    );
  }
}
```

Components im Detail

Arrays in States

```
class Familie extends React.Component {
  constructor() {
    this.state = {
      nachbarn: [
        "Jobs",
        "Gates"
        // "Musk" zieht ein
      ]
    };
  }

  render() { … }
}
```

Components im Detail

Arrays in States

```
this.setState({
  nachbarn: ["Musk"]
});
```

?

Components im Detail

Arrays in States

```
this.setState({
  nachbarn: this.state.nachbarn.push("Musk")
});
```

?

Components im Detail

Arrays in States

```
this.setState({
  nachbarn: [ ...this.state.nachbarn, "Musk" ]
});
```

Spread Operator (ES6)

?

Components im Detail

Arrays in States

```
this.setState({
  nachbarn: this.state.nachbarn.concat("Musk")
});
```

Klont das Array (ES5)

?

Components im Detail

Arrays in States

`setState()` setzt immer einen komplett neuen State!

Vorteile

Undo / Redo
Debugging
Aufzeichnung
Deterministisch

Faustregel

State als *immutable* betrachten!

Components im Detail

Arrays in States

```
class Familie extends React.Component {
  constructor() {
    this.state = {
      nachbarn: [
        "Jobs",
        "Gates", // neu: "Ballmer"
        "Musk"
      ]
    };
  }
}
```

Components im Detail

Arrays in States

```
const newState = this.state.nachbarn.slice();

newState[1] = "Ballmer";

this.setState({
  nachbarn: newState
});
```

Klont das Array

Components im Detail

React Hooks

„Stateless Components mit State“

Components im Detail

React Hooks

```
function Seconds() {
  // Kein this.state?!

  return <span>{this.state.name}</span>;
}
```

Components im Detail

React Hooks

```
function Seconds() {
  const [ second, setSecond ] = React.useState(0);

  return <span>{second}</span>;
}
```

Components im Detail

React Hooks

```
function Seconds() {
  const [ second, setSecond ] = React.useState(0);

  setInterval(() => setSecond(second + 1), 1000);

  return <span>{second}</span>;
}
```

Components im Detail

React Hooks

```
function Seconds() {
  const [ second, setSecond ] = React.useState(0);

  setInterval(() => setSecond(second + 1), 1000);

  // componentDidMount() / componentDidUpdate()

  return <span>{second}</span>;
}
```

Components im Detail

React Hooks

```
function Seconds() {
  const [ second, setSecond ] = React.useState(0);

  setInterval(() => setSecond(second + 1), 1000);

  // componentDidMount() / componentDidUpdate()
  React.useEffect(() => {
    // …
  });

  return <span>{second}</span>;
}
```

Components im Detail

React Hooks

```
function Seconds() {
  const [ second, setSecond ] = React.useState(0);

  setInterval(() => setSecond(second + 1), 1000);

  // componentDidMount() / componentDidUpdate()
  React.useEffect(() => {
    // …
    // componentWillUnmount()
  });

  return <span>{second}</span>;
}
```

Components im Detail

React Hooks

```
function Seconds() {
  const [ second, setSecond ] = React.useState(0);

  setInterval(() => setSecond(second + 1), 1000);

  // componentDidMount() / componentDidUpdate()
  React.useEffect(() => {
    // …
    // componentWillUnmount()
    return () => clearInterval(…);
  });

  return <span>{second}</span>;
}
```

Components im Detail

React Hooks

React Hooks = `useState()` + `useEffect()`

Tipps & Tricks

React typisiert: TypeScript

```
create-react-app my-app --typescript
```

Tipps & Tricks

React typisiert: TypeScript

```
interface Props {
  name: string;
  anzahl: number;
};
```

Tipps & Tricks

React typisiert: TypeScript

```
interface Props {
  name: string;
  anzahl: number;
};

class Familie extends React.Component<Props> {
}
```

Tipps & Tricks

Websockets leicht gemacht

```
class Stream extends React.Component {
}
```

Tipps & Tricks

Websockets leicht gemacht

```
class Stream extends React.Component {
  constructor() {
    this.state = { time: "" };
  }
}
```

Tipps & Tricks

Websockets leicht gemacht

```
class Stream extends React.Component {
  constructor() {
    this.state = { time: "" };
  }

  render() {
    return <div>{this.state.time}</div>;
  }
}
```

Tipps & Tricks

Websockets leicht gemacht

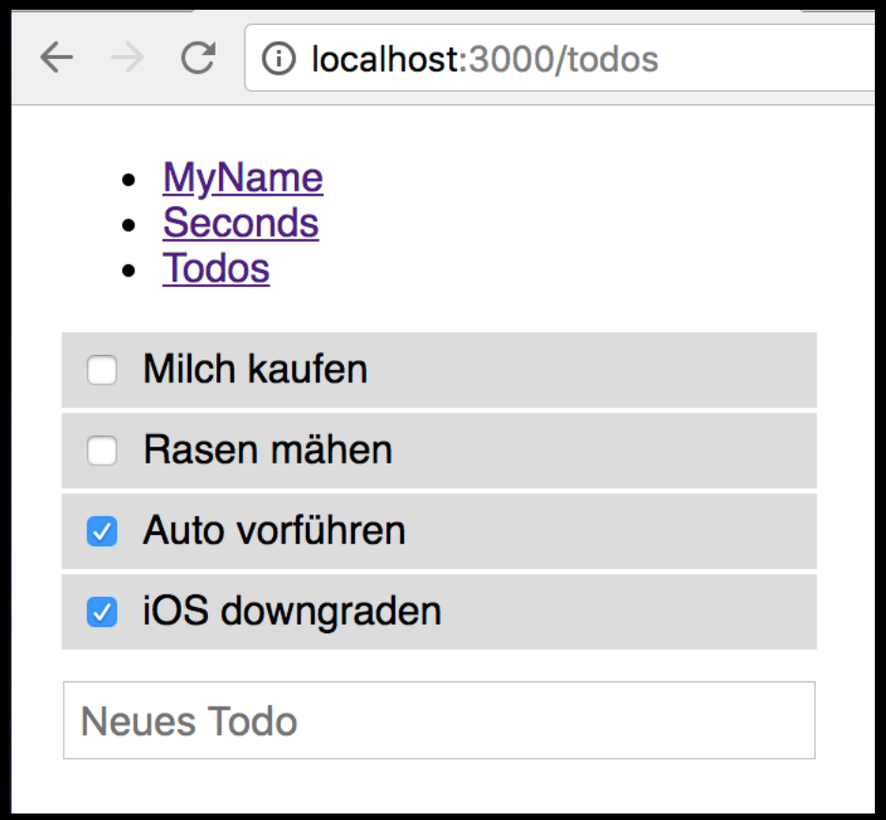
```
class Stream extends React.Component {
  constructor() {
    this.state = { time: "" };
  }

  render() {
    return <div>{this.state.time}</div>;
  }

  componentDidMount() {
    io("http://time.ch").on("time", time => {
      this.setState({ time });
    });
  }
}
```

Praxisübung

Todo-App



1. Todos anzeigen
2. Erledigte markieren können
3. Hinzufügen (keine leeren!)
4. Neue oben, erledigte unten
5. Tests schreiben